# Log-based Intrusion Detection for MANET

Mouhannad Alattar[1], Françoise Sailhan[2] and Julien Bourgeois[1]
[1] LIFC, University of Franche-Comté,25201 Montbéliard, FRANCE. [2] Cédric Laboratory, CNAM,Paris, FRANCE.

***Abstract*—Ad hoc networks operate mostly over open environments and are hence vulnerable to a large number of threats. This calls for providing advanced intrusion detection. To meet this requirement, we introduce IDAR, a signature- and log-based distributed intrusion detector dedicated to ad hoc routing protocols. Contrary to existing systems that observe packets, IDAR analyses the logs generated by the OLSR protocol and identifies patterns of misuse. This detector copes with the resource-constraints of devices by providing distributed detection. In particular, depending on the level of suspicion/gravity involved, in-depth cooperative investigation is launched. Simulation shows limited bandwidth usage, high detection and low false positives.**

***Index Terms*—Intrusion detection, MANETs, routing protocols, misuse.**

## I. INTRODUCTION

Securing *ad hoc* networks is challenging because these networks rely on an open medium of communication. In addition, they are cooperative by nature and hence lack of centralized security enforcement points, e.g., routers, from which preventive strategies are launched. Thus, traditional ways of securing networks relying on e.g., firewall, should be enriched with reactive mechanisms, e.g., intrusion detection system. Towards this goal, we survey the attacks targeting the Optimized Link State Routing (OLSR) routing protocol [1]; its central role consisting in determining multi-hops paths among the devices, designates this protocol as one of the favorite targets for attackers. We detail each attack, relying on a formalism that captures the complexity and temporal dependencies between each of the constituting sub-tasks. While describing an attack, we attempt to circumvent the general form of this attack so as to keep to a minimum intrusion detections that fail due to slightly varying attacks. Based on these modeled attacks, we further implement one attack to which our detection system is challenged. Recent works show that intrusion may be identified as a deviation of the correct behavior (anomaly detection); this correct behavior is either hand-specified relying on a protocol description, e.g., [2] or automatically built/analyzed using machine learning or data mining techniques, e.g., [3]. The difficulty inherent in automatically modeling the behavior of dynamic routing protocols leads to many false positives, which can be reduced by coupling automatic and specification-based anomaly detection [4]. An alternative describes the way the intruder penetrates the system (by establishing intrusion signature) and detects any behavior that is close to that signature. Little attention - to the best of our knowledge, only couple of works [5], [6] - focuses on signature-based detection in *ad hoc* networks.

We propose IDAR, a signature-based Intrusion Detector dedicated to Ad hoc Routing protocols. The general idea lies in taking advantage of the audit logs that are generated by the routing protocol so as to detect intrusion attempts. While not requiring changes in the implementation of the routing protocol, IDAR does not necessitate inspecting the traffic as it is the case with other (aforementioned) systems. Main challenges come from the need to keep to a minimum the number of investigations along with the computational load related to the identification of intrusions. This calls for proposing a lightweight intrusion detection that copes with the cooperative nature of *ad hoc* networks and the resource constraints of mobile devices. Towards this goal, we propose a distributed and cooperative intrusion detection system that parses log as close as possible to the device that generates it so as to diminish remote communications. Based on the parsed logs, intrusion detection takes place. This consists in identifying patterns of events that characterize intrusion attempts. In practice, a sequence of relevant events are extracted from logs and are matched against intrusion signatures. In order to minimize the number of investigations, events are categorized. Then, depending on their level of criticality, distributed and cooperative investigation might be conducted. We further evaluate the performance of the proposed system.

The remainder of this paper is organized as follows. We first survey attacks on *ad hoc* routing protocols (§II). Based upon the defined intrusion signatures, we present IDAR (§III) and evaluate its performance (§IV). Then, we conclude with a summary of our results along with directions for future works (§V).

## II. VULNERABILITIES

*Ad hoc* routing protocols constitute a key target for attackers because: (i) no security countermeasure is specified/implemented as a part of the published RFCs, (ii) the absence of a centralized infrastructure complicates the deployment of preventive measures e.g., firewalls, and (iii) devices operate as routers, which facilitates the manipulation of messages and more generally the compromising of the routing. We hereafter illustrate our presentation by exemplifying attacks on a proactive protocol, OLSR [1].

### A. Background on OLSR

OLSR aims at maintaining a constantly updated view of the network topology on each device. One fundamental principle is the notion of multipoint relay (MPR): each device selects a subset of 1-hop neighbors, the MPRs, that are responsible for forwarding the control traffic. The idea is to select the minimum number[1] of MPRs that cover 2-hops neighbors so as to reduce the number of nodes retransmitting messages and hence keep to a minimum the bandwidth overload. In practice, a node $N$ selects MPRs among the 1-hop neighbors that are announced in periodic heartbeat messages, termed *hello* messages. Then, a *Topology Control* (TC) message intended to be diffused in the entire network, is created by the selected MPR(s). In this message, a MPR declares the nodes (including $N$) that selected itself to act as a MPR. Then, any device can compute the shortest

[1]Redundant MPRs may be selected to increase the availability.

TABLE I
NOTATIONS

| **Communication** | | | |
|---|---|---|---|
| $Y \xrightarrow{CM_t} X$ | At $t$, $Y$ sends a control message $CM$ that is received by $X$ | | |
| $Y \xrightarrow{CM_t}$ | $Y$ sends a control message CM at $t$ | $Y \not\xrightarrow{CM_t}$ | $Y$ does not send a control message CM at $t$ |
| $\xrightarrow{CM_t} X$ | $X$ receives a control message CM at $t$ | $\not\xrightarrow{CM_t} X$ | $X$ does not receive a control message CM at $t$ |
| **Parameters & Messages** | | | |
| $CM$ | Control Message | $\triangle t$ | Period of time |

path, represented as a sequence of MPRs, to any destination. In addition, recent versions of the specification support a node holding several network interfaces which are declared (if many) in a so-called MID (Multiple Interface Declaration) message. This message is broadcasted regularly by MPRs so that one another maps multiple interfaces with a main address. Thus, a unique identification is provided. Additional extensions have been devised in compliance with the above-summarized core functions. Examples include (i) dealing with the nodes that commit (or not) to carry the traffic for others, and (ii) supporting the interconnection of an OLSR MANET with another routing domain. With the former, nodes advertise their willingness to carry/forward traffic. With the latter, OLSR is extended to import (and resp. export) the routes provided by other routing protocols (resp. OLSR). In particular, any gateway with associated host(s) and/or network(s) generates periodically a HNA (Host and Network Association) message including those host(s) and/or network(s) (i.e., the related network address and netmask); this message being further disseminated by MPRs. Overall, these core and auxiliary functionalities are together subject to various attacks.

## *B. Attacks Targeting OLSR*

OLSR is vulnerable to a wide range of attacks, which are hereafter sub-classified according to the action which is undertaken on the routing [7]:

- *Drop attacks* consist in dropping routing message(s).
- *Active forge attacks* generate novel and deceptive routing message(s).
- *Modify and forward attacks* modify a received routing message(s) before forwarding it.

These above attacks are further formulated according to the model introduced in [8]. This model specifies the relationship between actions that compose an attack as well as their related consequences. We further enrich this model with temporal annotations (Table I). Therefore, complex attacks, their constituting actions and consequences are temporally depicted and categorized.

**Drop attack** comes from a node that drops a message instead of relaying it. Threatened messages are restricted to the messages that are created and relayed by MPR, i.e., TC, MID, and HNA messages. Consider a host $H$ that sends a message which intended to be forwarded. This message is received by $I$ that drops it. In practice, $I$ drops a message if $I$ does not forward it within the maximum allowed period. Convenient drop is due to a packet that is empty, expired (as indicated by the time to live field), duplicated or out of sequence. In addition, restrictive forwarding may apply; only the MPR(s) forward messages. Otherwise, remaining drops come from either a selfish/faulty node or an attacker. An attempt to drop any packet is termed black hole whereas selective dropping is named gray hole. Rather than dropping traffic, an opposite behavior consists in introducing falsified routing information.

**Active forge** comes from a node that injects novel and deceptive routing messages. Among others, the broadcast storm aims at exhausting resources (e.g., energy). For this purpose, an intruder $I$ forges a large number of control messages $CM$ within a short period of time. This attack may be conducted in a distributed manner with several nodes colluding so as to emit (a large number of) messages. In order to reduce the visibility of this attack, $I$ typically masquerades itself. In practice, masquerading lies in sending a message CM including a switched identification $S$ ($I \xrightarrow{CM(I)}, I \xrightarrow{CM(S)}$). Note that this case should be distinguished from a node that holds several interfaces and advertises them in the dedicated MID message. Apart from masquerading, identity spoofing may be intended to create conflicting route(s) and potentially loop(s). This spoofing attack may also be coupled with a modification of the willingness field so as to impact the selection of MPR. MPRs are selected among the nodes with the highest willingness and in case of multiple choices, the node providing a reachability to the maximum number of 2-hops nodes is primarily selected. For instance, a node whose willingness attribute set to $will\_never$ (resp. $will\_always$), is never (resp. always) selected as a MPR. In addition, active forges cover the message tempering with incorrect adjacent links (*hello* messages), topology information (TC messages), and network interfaces (MID) and routes (HNA messages). With the former, $I$ forges at $t'$ a *hello* message, which declares 1-hop and symmetric neighbors $NS'_I$ differing from the real one $NS_I$: $I \xrightarrow{hello(NS'_I)_{t'}} S, NS'_I \neq NS_I \Rightarrow I \in \mathcal{I}$. When forging $NS'_I$, the attacker has 3 options:

- declaring a non-existing node as a symmetric neighbor, implies that $I$ (or another misbehaving node) is further selected as a MPR (Expression 1): if $I$ advertises a non-existing node $N$ ($N \notin \mathcal{N}$ with $\mathcal{N}$ defining the set of nodes composing the OLSR network), $I$ ensures that no other (well-behaving) MPR claims being a 1-hop symmetric neighbor of $N$. Recall that MPRs are selected so that all the 2-hops and symmetric neighbors are covered, $I$ is selected as a MPR.

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t}, I \xrightarrow{hello(NS'_I)_{t'}} S, |t'-t| < \triangle t, \\ \exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I \\ \Downarrow \\ I \in \mathcal{I}, \exists I' \in \mathcal{I} \cap NS_S \ni: I' \in MPR_S, \\ Card(NS'_I \backslash NS'_I \cap \mathcal{N}) > 0. \end{gathered} \tag{1}$$

This is verified as long as no other misbehaving neighbor of $S$ claims the same. Overall, inserting at least one non-existing neighbor ($\exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I$) guaranties that a misbehaving node $I'$ (with $I' \in \mathcal{I}$) is selected to act as a MPR of $S$ ($\exists I' \in \mathcal{I} \cap NS_S \ni: I' \in MPR_S$). In addition to the above, the connectivity of $I$ increases.

- declaring that an existing node is a symmetric 1-hop neighbor whereas it is not the case ($\exists X \in NS'_I \cap \mathcal{N} \ni: X \notin NS_I$). This claiming increases artificially the connectivity of $I$, i.e., $Card((NS'_I \backslash [NS'_I \cap NS_I]) \cap \mathcal{N}) > 0$. If no (well-behaving) MPR covers $S$ ($\nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in NS_S \wedge X \in NS_A$), then at least one misbehaving node is selected as a MPR of $S$ ($\exists I' \in \mathcal{I} \ni: I' \in MPR_S$). This typically characterizes an attempt to create a blackhole: $I$ introduces a novel path that provisions the blackhole.

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t}, I \xrightarrow{hello(NS'_I)_{t'}} S, |t'-t| < \triangle t, \\ \exists X \in NS'_I \cap \mathcal{N} \ni: X \notin NS_I \\ \Downarrow \\ I \in \mathcal{I}, \\ Card((NS'_I \backslash [NS'_I \cap NS_I]) \cap \mathcal{N}) > 0, \\ \nexists A \in \mathcal{N} \backslash \mathcal{I} \ni: A \in N_S \wedge X \in N_A \\ \Downarrow \\ \exists I' \in \mathcal{I} \ni: I' \in MPR_S. \end{gathered} \tag{2}$$

- omitting an existing 1-hop symmetric neighbor $P$ ($\exists P \in NS_I \ni: P \notin NS'_I$), decreases artificially the connectivity of both $P$ and $I$ ($NS_I \nsubseteq NS'_I$):

$$\begin{gathered} S \xrightarrow{hello(NS_S)_t}, I \xrightarrow{hello(NS'_I)_{t'}} S, \\ |t'-t| < \triangle t, \exists P \in NS_I \ni: P \notin NS'_I \\ \Downarrow \\ I \in \mathcal{I}, \exists I' \in \mathcal{I} \cap NS_S, NS_I \nsubseteq NS'_I. \end{gathered} \tag{3}$$

Overall, such a falsification of the neighboring adjacency perverts the topology seen by $S$ and may impacts the selection of MPR(s) of $S$. Another alternative refers to a node $I$ declaring itself as a MPR although it has not been selected as a MPR forehand: $I$ forges a (TC) message including incorrect 1-hop symmetric neighbor(s), including at least the MPR selector(s) that corresponds to the neighbors that have selected $I$ as a MPR. Let $\mathcal{A}_I$ represents this set advertised in the TC message. This attack consists in $I$ advertising an incorrect $\mathcal{A}'_I$ differing from the real one $\mathcal{A}_I$: $I \xrightarrow{TC(\mathcal{A}'_I)_t} S, \mathcal{A}'_I \neq \mathcal{A}_I \Rightarrow I \in \mathcal{I}$. In particular, possible falsifications lie in inserting a non-existing node or an incorrect but existing node or also omitting a node $S$ belonging to $\mathcal{A}_I$. Due to the lack of space, we do not detail herein each of these cases. Upon the reception of a falsified TC message, routing tables are corrupted and may contaminate any interconnected routing domain if a gateway exports those OLSR routes. Note that the gateway may also forge itself wrong routes. This attack constitutes a generalization of the previously-defined forging of corrupted TC messages: a node advertises either non-existing or existing but unreachable nodes, or omitting advertising reachable nodes. Symmetrically, an intruder may import incorrect routes to the OLSR domain. Overall, the forge attacks (e.g., route spoofing attacks) necessitate to tamper message while keeping it syntactically correct. In other words, bogus messages can be forged, hence creating implementation-dependent effects. Generally speaking, similar tampering may be performed by a MPR relaying control messages.

**Modify and forward attacks** are characterized by an intermediate that captures the control message and replays or/and modifies this message before forwarding it. Replaying a message includes delaying (i.e., forwarding latter potentially in another area) and repeating this message. As a result, routing tables are updated with obsolete information. Both attacks can be performed in a distributed manner with two intruders: one recording the message from one region so as to replay it in another region (i.e., the one of the colluding intruder); this leads to the creation of a *wormhole*. In order to stay invisible, both intruders may keep the identification field unchanged: the source is still $S$. Note that sequence numbers constitute a standard mechanism that provides protection against replay attacks. Based on those numbers, a node identifies freshest information, prevents duplicates and replaying while indicating insertion/deletion. In counterpart, there usage may be hijacked. For instance, an intruder $I$ may forwards the message including an increased sequence number. Thus, the source assumes that $I$ provides the freshest route.

## III. Intrusion Detection

In order to deal with such attacks, we propose IDAR, a distributed, log- and signature-based intrusion detection system that periodically collects the OLSR logs. These logs characterize the activities of OLSR (e.g., packet reception, MPR selection). Note that additional logs, e.g., system-, security-related logs, could be integrated and correlated. Once parsed, a log is used so as to detect a sign of suspicious activity. This consists in matching the log against predefined signatures; a signature is thought as a partially ordered sequence of events that characterizes a misbehaving activity. Detection is potentially not only a memory but also a bandwidth-consuming: it may involve not only examining logs but also requesting others to collect/correlate additional intrusion evidences. Thus, this activity should be carefully-planned, i.e., initiated only when sufficient suspicion exists and terminated as soon as a result is obtained. Toward this goal, evidences are classified so that depending on their level of gravity, additional in-depth detection might be performed. They fall into the following groups:
- *Suspicious-evidence-group* contains the evidences necessary to identify a node as suspicious,
- *Initial-evidence-group* contains the evidences necessary to identify a suspicious node & launch a networked investigation,
- *Confirming-evidence-group* contains the evidences that confirm the occurrence of an attack. This results in terminating the investigation and declaring the suspicious node as an intruder.
- *Canceling-evidence-group* contains the evidences that eliminate the suspicion, which ends the investigation.

These groups are populated with the evidences extirpated from logs. If an evidence belonging to the initial-evidence-group is discovered then an advanced investigation is launched so as to confirm (Confirming-evidence-group) or infirm (canceling-evidence-group) intrusion; both resulting in terminating the investigation. Relying on these groups, the gradual evolving of the attack and of its related detection are easily followed. In addition, its compact form facilitates the lightweight discovering of long-terms intrusions. But before delving into the functioning of the above groups, let first exemplify the proposed intrusion detection system with the link spoofing attack we purposely developed.

## A. Link Spoofing Signature

Link spoofing lies in falsifying *hello* message(s) so as to modify the topology perceived by adjacent nodes and confer to the attacker the ability to isolate node(s) and/or the MPR position permitting to e.g., mis-relay. The link spoofing (Expression 4) we developed takes the following form:

- non-existing and symmetric node(s) ($\exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I$) are advertised by the intruder $I$. This attack guaranties[2] that $I$ is selected as a MPR because $I$ is the only MPR that covers the non-existing node[3].
- advertises existing but non-neighboring node(s). In particular, $I$ announces common neighbors with another node $L$ adjacent to $S$, $I$ increases the probability that $L$ is not selected as a MPR ($I$ replacing $L$) and henceforth increases its proper ascendancy.
- does not advertise neighboring and symmetric node(s). A drawback is that it decreases the connectivity of the malicious MPR, which does not facilitate its selection. For this purpose, we ignored this case although the signature deals with it. This attack decreases the connectivity of node(s).

$$\begin{gathered} S \xrightarrow{Hello(NS_S)_t} I, I \xrightarrow{Hello(NS'_I)_{t'}} S, |t'-t| < \triangle t, \\ \exists N \in NS'_I \ni: N \notin \mathcal{N} \cap NS_I, \\ \exists L \in MPR_S \ni: [NS_L \backslash NS_I] \subseteq [NS'_I \backslash NS_I] \\ \Downarrow \\ I \in \mathcal{I}, L \notin MPR_S, \\ \exists I' \in \mathcal{I} \ni: I' \in MPR_S, \\ Card(NS'_I \backslash [NS_I \cap \mathcal{N}]) > 0, \\ Card(NS'_I \backslash NS_L) > 0. \end{gathered} \tag{4}$$

## B. Cooperative Investigation

Link spoofing aims at inflecting the MPR selection; such selection is triggered upon a change in the symmetric 1- and 2-hops neighbors. Rather than launching an in-depth investigation upon these changes, we minimize the investigation by initiating it only when the event that occurs is relevant to a link spoofing attack. We ignore changes in the 1-hop neighborhoods (e.g., node apparition) because they are observed by the node itself and are hence not subject to remote falsification; a cornerstone of a link spoofing. In addition, changes in the 2-hops neighborhood are considered if they impact the MPR selection. Those include:

- A replaced MPR (Evidence 1, $E1$ for short) means that a change in the covering of 1-hop neighbors leads to this replacement. This comes from 1-hop neighbor(s), possibly the replacing MPR, that increase(s)/decreases it/their coverage(s) to the detriment of the replaced MPR.
- No MPR replacement takes place but a previously-selected MPR is detected as misbehaving. Messages are dropped, forged or misrelayed by that MPR ($E2$). Overall, a link spoofing also covers the case wherein an intruder continues to advertise identical 1-hops neighbors despite recent changes in the neighborhood. Note that contrary to the other evidences listed here, this case is not event-driven and should be handled by launching periodical/random checks.
- a MPR is the only one providing the connectivity to node(s) ($E3$).
- a MPR does not cover its adjacent neighbor(s) ($E4$).
- a MPR provides connectivity to a non-neighbor ($E5$).

$$\begin{array}{cc} (E1 \vee E2)\ , & \text{optional}(E3) \\ \Downarrow & \Downarrow \\ E4 \vee E5 & (!E4 \wedge !E5) \\ \Downarrow & \Downarrow \\ \text{The suspicious MPR} & \text{The suspicious MPR} \\ \text{is an intruder.} & \text{is well-behaving.} \end{array} \tag{5}$$

The occurrence of $E1$ or $E2$ is the starting point for further investigation; $E1$, $E2$ hence belong to the initial-evidence-group. Note that a MPR that is the only one providing the connectivity to node(s) ($E3$) is suspicious but this condition is not sufficient to launch an investigation: (i) this situation is typical in a sparse network and (ii) 2 nodes within communication range often fail in communicating due to the unpredictable nature of wireless transmission resulting from, e.g., obstacles, noises. Thus, diagnosing $E3$ is especially difficult under no specific assumption. Overall, the occurrence of either $E1$ or $E2$ and optionally $E3$ leads to an in-depth investigation. In practice, the investigator interrogates the 1-hops neighbor(s) of the suspicious MPR so as to discover whether the suspicious MPR does not cover its neighbors ($E4$) or advertises a distant node ($E5$). If all the requested nodes confirm (resp. infirm) $E4$ or $E5$ , then the MPR is suspected (resp. well-behaving). Note that, if part of those requested nodes express a different opinion, which may result from nodes' mobility or misbehavior(s), the number of these nodes and their reputation should be taken into account (as we plan in our near future work). More precisely, this

**Algorithm 1** : Advanced Investigation

```
1: SuspiciousMPRs= new (MPR)
2: OldMPRs = GetReplaced-MPR();
3: for ($suspicious \in SuspiciousMPRs$) do
4:     Common2HopsNeighbors = GetCommon2HopsNeighors(suspicious, OldMPRs )
5:     for ($2HopsNeighbor \in Common2hopsNeighbors$) do
6:         if ($VerifyLink(2HopsNeighbor, suspicious) == false$) then
7:             GenerateAlarm(suspicious);
8:             Terminate(suspicious);
9:         end if
10:    end for
11:    Cancel(suspicious);
12:    SuspiciousMPRs=SuspiciousMPRs - suspicious;
13: end for
```

investigation (Algorithm 1) is conducted as follows: replacing MPR(s) and replaced MPR(s) are computed (lines 2, 3); the 2-hop neighbors that are covered by both are established (line 4) so as to be interrogated. In practice, this interrogation of a 2-hops neighbor $Ai$ consists in sending a request to $Ai$ without going through both the suspicious MPR $I$ or a colliding intruder $I'_j$. This avoidance is necessary so as to prevent $I$ and $I'_j$ from dropping the request and/or simply forging a defective answer. For this purpose, a 1-hop neighbor (primarily the MPR) that

[2]unless another attacker advertises the same non-existing node.

[3]With last version of RFC, a node (that is potentially an intruder) can dictate its selection as MPR by advertising its willingness to relay messages for others.

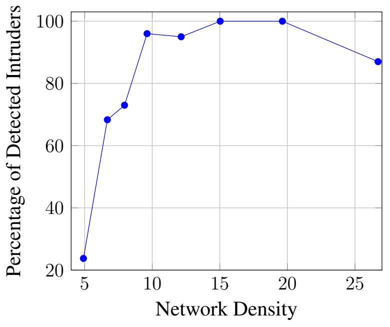
covers the requested 2-hops neighbors is provided the request. If no answer is obtained (i.e., when the related time-out elapses), then the demand is sequentially transferred through the rest of the covering 1-hop neighbors (as aforementioned, MPRs being primarily selected). Note that this verification is performed within a thread so that the investigation of one node (and the result waiting) is not blocking to others. If no neighbor is left, then a (multi-hops) alternative path is researched in the routing table to reach $Ai$. If $Ai$ denies the suspicious MPR as 1-hop neighbor then an alarm is generated and the overall investigation terminates. Otherwise, if no deny is provided, the suspicious MPR is well behaving. Note that if no answer is provided about the suspicious MPR, then the suspicious MPR is tagged as not verified.

## IV. Performance Evaluation

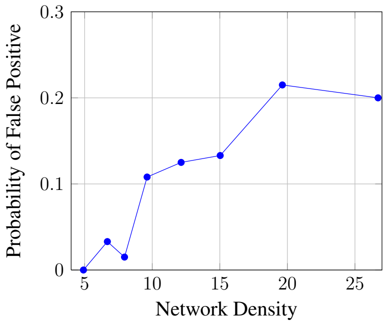
In order to evaluate the performance of IDAR, we simulated a mobile *ad hoc* network using the NS3 simulator[4]. Each node constituting this simulated network is further virtualized relying on a LinuX Container virtual machine[5] that provides an operating-system-level virtualization; permitting to run multiple isolated nodes on a single host and hence, to independently measure the memory usage. Relying on this platform, simulations are carried for a duration of 120s on Intel(R) Core(TM) Duo 2.53GHz with each simulated node holding a fedora 12 operating system. We consider a MANET of $N = 25$ nodes split into 20 well-behaving nodes and 5 intruders, together communicating *via* IEEE 802.11a with a transmission range $T_x$ of 175m. They move, relying on the so-called $RandomWalk2d$ mobility pattern provided by NS3: nodes randomly choose directions, and herein move at the same speed. When a node hits the network boundaries (unless specified, the network area is defined by $S_x \times S_x$ =400m $\times$ 400m), it rebounds in a reflexive angle. We use OLSR preserving the parameters promoted in the RFC. Data traffic is further simulated relying on the V4PingHelper of NS3: each node exchanges 56 bytes echo to one another and waits 1s before sending it again. We evaluate the performance of IDAR in terms of intruder detection rate, probability of false positive and detection overhead; the traffic and memory usage dedicated to intrusion detection. Performance indicators are evaluated with regard to the network density (that reflects the scaling properties) and the mobility. The node density corresponds to the average number of neighbors defined by $\frac{N\times\pi\times T_x^2}{S_X^2}$ in [9]. Figure 1-(a) shows that the percentage of detected intruders greatly increases until it stabilizes to a high detection rate (around 96%) and decreases for a density higher than 21 neighbors. This decrease comes from a high collision rate which prevents nodes from efficiently both investigating intrusion and receiving data/routing traffic. The probability of false positive keeps low but rises according to the network density (Figure 1-(b)). This comes from two nodes owning different live-times for a bidirectional link connecting each other. This can be overcome by amplifying the period during which those links are considered valid [10]. Varying the **mobility** (Figure 2) and locking the network density to 15 neighbors, the percentage of detected intruders and false positives are quite stable: around 95% of the intruders are

[4]http://www.nsnam.org

[5]http://lxc.sourceforge.net



(a)



(b)

Fig. 1. Percentage of Detected Intruders and False Positive Rate Depending on the Network Density

detected with a velocity lower than 8 m/s (i.e., around 28.7 km/h). Then, the detection rate declines and the false positives rate increases. This comes from the high turnover (and the resulting losses of connectivity) that prevents from performing full investigation. Such a high mobility not only prevents from detecting intrusion but also from communicating over multi-hops. Additional experiments show that the traffic generated by IDAR rises according to the network density. That results from the collisions that imply to re-launch investigation. Nevertheless, the traffic generated by our system keeps negligible comparing to the OLSR: around 1.1% (and resp. 11.4%) of the OLSR traffic for a network density equals to 5 (resp. 10); this leads to a maximum memory usage corresponding to 8.45 MB (and resp. 19.08 MB).

## V. Conclusion

Signature- and specification-based detection are unpopular comparing to anomaly detection. This calls for consolidating efforts on signature- and specification-based detection while following the *habitus* that lies in coupling detection systems together. To meet this requirement, we define the signatures of the attacks targeting OLSR. These signatures are utilized by IDAR, a log-based, distributed intrusion detection system. IDAR distinguishes itself by analyzing the logs generated by a routing protocol and extracting intrusion evidences in order to compare these latter against predefined intrusion signatures. For this purpose, evidences are categorized into 4 groups according to their degree of suspicion/gravity and hence to their ability to activate/deactivate the investigation. We further develop a link spoofing attack, build the related detection rules, and evaluate the performances of IDAR relying on the NS3 simulator coupled

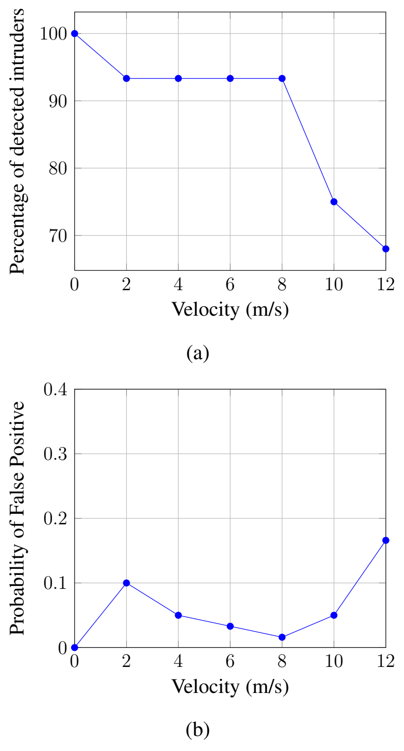


Fig. 2. Percentage of Detected Intruders and False Positive Rate According to the Mobility.

with virtual machines. Overall, experiments show a high rate of intrusion detection and low false positive rate even under increased mobility and density. Meanwhile, memory and bandwidth consumption are adapted to resource-constrained devices. Still, these experiments show that intrusion detection induces additional activities, hence calling for finding a compromise between detection accuracy and reduced resource-consumption. In addition, we envisage coupling IDAR with trust model in oder to avoid the bad effects of deceptive opinion during the investigation.

## VI. Related Work

Systems that detect intrusions targeting *ad hoc* routing protocols are diverse in the way they analyze the intrusion. They fall into 3 categories: anomaly, specification- and signature-based detection. **Anomaly detection** constitutes the main approach. The idea is to define the correct behavior of a node and detect deviations from this behavior. This correct behavior is automatically built during an attack-less phase. In [11], attempts to falsify the routes are detected. During the training phase, the impact of the movement on the percentage of updates in the routing table is analyzed. Then, during operation, an actual percentage of updates differing from the predicted one, is defined as an anomaly; the distinguish is provided by the Support Vector Machine (SVM) Light [12] classifier or a rule-based engine RIPPER [3] . In [13] and resp. [14], a blackhole (and resp. dropping) attack targeting the AODV protocol (resp. a secured version of AODV) are detected by investigating features, e.g., the number of route requests and route replies as well as the average difference of sequence numbers[6]. If the distance between observed features and the average ones, exceeds a given threshold, then an intrusion is detected. More sophisticated Cross-Features Analysis (CFA) [15] is applied relying on the C4.5 [16] decision tree classifier so as to detect both blackhole and packet dropping on AODV and DSR protocols. CFA and C4.5 are also used for OLSR [17]. Rather than establishing automatically a correct behavior, **specification-based** systems hand-code this behavior relying on the protocol specification. Then, the system detects a violation of constraints circumventing this behavior. Example of constraints defining the correct behavior of OLSR[2], [4] includes the fact that a MPR and a node that selects the MPR must be adjacent. These constraints are modeled using semantic properties [4], rules [2], or finite state machines [10]. **Signature-based detection** models the way an intruder penetrates the system by defining intrusion signatures. Then, any behavior that is close to this predefined signature is flagged as intrusion. Finite-state-machine is used to detect network flooding, dropping and spoofing attacks, which target AODV [6]. Sensors observe the traffic and match it against predefined signatures. They also exchange MAC and IP addresses to detect identity spoofing realized by by a node emitting a packet identified with MAC or IP addresses differing from those registered. Rule-based signatures are specified to detect attack on OLSR [5] in opposition to the legitimate behavior depicted by a prior specification-based IDS [4]. This detection is further coupled with a trust system: a node mistrusts another that does not conforms to the predefined rules.

[6]Increased sequence numbers are known as a sign of blackhole attack.

## References

[1] T. Clausen and P. Jacquet, “Optimized link state routing protocol (OLSR),” IETF experimental RFC 3626, 2003.
[2] F. Cuppens, N. Cuppens-Boulahia, S. Nuon, and al., “Property based intrusion detection to secure OLSR,” in *IEEE ICWMC*, 2007.
[3] W. Cohen, “Fast effective rule induction,” in *ICML*, 1995.
[4] M. Wang, L. Lamont, P. Mason, and al., “An effective intrusion detection approach for OLSR MANET protocol,” *IEEE ICNP workshop*, 2005.
[5] A. Adnane, R. Sousa, C. Bidan, and al., “Autonomic trust reasoning enables misbehavior detection in OLSR,” in *ACM SAP*, 2008.
[6] G. Vigna, S. Gwalani, K. Srinivasan, and al., “An intrusion detection tool for AODV-based ad hoc wireless networks,” in *ACSAC*, 2004.
[7] N. Peng and S. Kun, “How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols,” *Ad Hoc Network*, vol. 3, no. 6, 2005.
[8] A. Adnane, C. Bidan, and R. Junior, “Trust-based countermeasures for securing olsr protocol,” in *IEEE CSE*, 2009.
[9] J. Haerri, F. Filali, and C. Bonnet, “Performance comparison of AODV and OLSR in vanets urban environments under realistic mobility patterns,” in *IFIP Med-Hoc-Net workshop*, 2006.
[10] C. Tseng, S. Tao, P. Balasubramanyam, and al., “A specification-based intrusion detection model for OLSR,” in *RAID*, 2008.
[11] Y. Zhang and W. Lee, “Intrusion detection in wireless ad-hoc networks,” in *ACM MobiCom conference*, 2000.
[12] T. Joachims, *Making large-scale support vector machine learning practical*. MIT Press Cambridge, 1999.
[13] S. Kurosawa, H. Nakayama, N. Kato, and al., “Detecting blackholes attack on AODV-based mobile ad hoc networks by dynamic learning method,” *International journal of network security*, vol. 5, no. 3, 2007.
[14] L. Bononi and C. Tacconi, “Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks,” *International journal of information security*, vol. 6, 2007.
[15] Y. Huang, W. Fan, W. Lee, and al., “Cross-feature analysis for detecting ad-hoc routing anomalies,” in *IEEE ICDCS*, 2003.
[16] J. Quinlan, *C4.5: programs for machine learning*. M. Kaufmann, 1993.
[17] J. Cabrera, C. Gutierrez, and R. Mehra, “Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks,” *Information Fusion*, vol. 9, 2008.